WHAT IS CLAIMED IS:

1.   A lock contention management method, comprising:

determining whether a code module should continually request access to a lock to keep other code modules from accessing a resource or should stop requesting access to the lock; and

if the code module should continually request access, allowing the code module to do so at a lowered priority.

2.   The method of claim 1, wherein:

the code module continually requests access to the lock when the resource has other tasks to run; and

the code module stops requesting access to the lock when the resource has no other tasks to run.

3.   The method of claim 1, further comprising determining that there are multiple processor run queues and having the code module stop requesting access to the lock if there are other code modules in the multiple processor run queues waiting to access the lock and having the code module continually request access to the lock if there are not.

4.   The method of claim 1, further comprising determining that there is a single processor run queue and having the code module stop requesting access to the lock if there are other code modules in the single processor run queue waiting to access the lock and having the code module continually request access to the lock if there are not.

5.   The method of claim1, wherein the code module stops requesting access to the lock using a low-priority execution technique that lowers a priority of the code module.

6.   The method of claim 5, wherein the low-priority execution technique allows a higher-priority code module to access the lock first.

7. The method of claim 6, wherein the lowered-priority code module continually requests access to the lock if no higher-priority code modules are available.

8. The method of claim 5, further comprising remembering an original
5 priority of the code module.

9. The method of claim 8, further comprising restoring a priority of the code module to the original priority.

10 10. The method of claim 9, wherein the original priority is restored after a specified period of time.

11. A lock contention management system, comprising:

a dispatch management module that determines when a code module
15 should wait for the lock by becoming undispatched and when the code module should try to access the lock by spinning; and

a low-priority execution module that lowers a priority of the code module when the code module becomes undispatched.

20 12. The lock contention management system of claim 11, wherein the dispatch management module determines that the code module should spin when a processor has no other tasks to perform.

13. The lock contention management system of claim 11, wherein the
25 dispatch management module determines that the code module should become undispatched when a processor has other tasks to perform.

14. The lock contention management system of claim 11, wherein the code module is a program thread.
30

15. The lock contention management system of claim 11, wherein the low-priority execution module reduces an original priority of the code module to a lowered priority.

16.   The lock contention management system of claim 15, wherein the low-priority execution module remembers the original priority of the code module and restores the original priority after a time.

5   17.   The lock contention management system of claim 15, wherein the low-priority execution module allows higher-priority code modules to acquire the lock before the lower priority code module.

18.   A method of acquiring a lock to allow execution of a program thread by a

10   computer processor, comprising:

determining whether to have the program thread to spin or become undispatched while waiting to acquire the lock; and

lowering a priority of the program thread before spinning or undispatching.

15

19.   The method of claim 18, wherein the program thread is allowed to spin if the computer processor does not have other processing to perform.

20.   The method of claim 18, wherein the program thread is allowed to

20   become undispatched if the computer processor has other processing to perform.